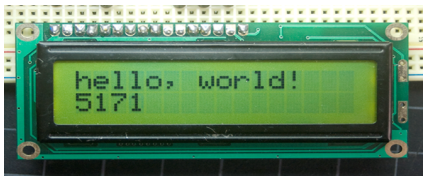
Βιβλιοθήκη για οθόνη LCD

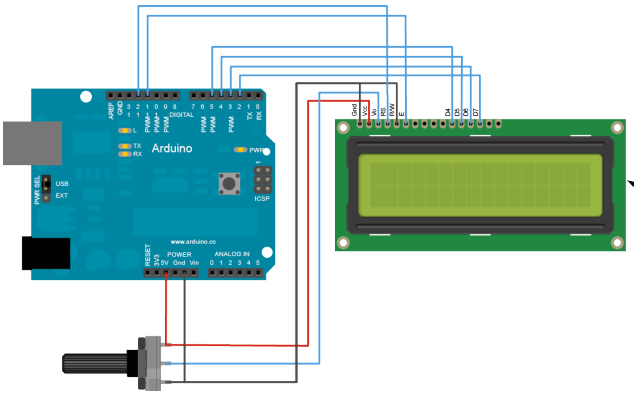
Η **βιβλιοθήκη LCD** βρίσκεται προεγκατεστημένη στο περιβάλλον προγραμματισμού του Arduino (IDE) και για την ορθή της λειτουργία ακολουθούμε την παρακάτω διαδικασία:



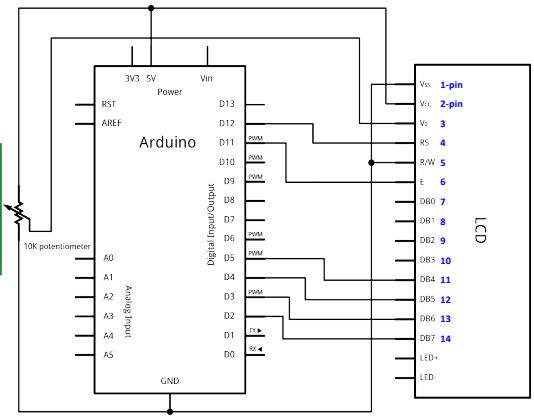
- Δηλώνουμε τη χρήση της βιβλιοθήκης στην αρχή του κώδικα π.χ. **#include <LiquidCrystal.h>**
- Δηλώνουμε τις θύρες (pin) στις οποίες θα συνδέσουμε την οθόνη π.χ.
  LiquidCrystal lcd(RS_pin, Enable_pin, Dpin4, Dpin5, Dpin6, Dpin7)
  για το συγκεκριμένο παράδειγμα έχουμε **LiquidCrystal lcd(12, 11, 5, 4, 3, 2)** (η δήλωση αυτή γίνεται πριν την συνάρτηση **setup()**).
- Χρησιμοποιώντας την εντολή **lcd.begin();** στην συνάρτηση **setup()** ορίζουμε τις γραμμές και τις στήλες της οθόνης που χρησιμοποιούμε, π.χ. **lcd.begin(αριθμός στηλών, αριθμός γραμμών),** η οθόνη που χρησιμοποιείται ως παράδειγμα μέσα στην βιβλιοθήκη έχει **16** στήλες και **2** γραμμές.
- Με την εντολή **lcd.print();** στην συνάρτηση **loop()** μπορούμε να εκτυπώσουμε οποιοδήποτε μήνυμα στην οθόνη π.χ. **lcd.print("hello, world!");**
- Τέλος με την εντολή **lcd.setCursor();** στην συνάρτηση **loop()** ορίζουμε την θέση του κέρσορα π.χ. **lcd.setCursor(0, 1);** θέτουμε τον κέρσορα στην πρώτη στήλη της οθόνης και στην δεύτερη γραμμή **αφού η αρίθμηση ξεκινά από το μηδέν.**

Παράδειγμα χρήσης της βιβλιοθήκης για οθόνη «LCD»

Παρακάτω παραθέτουμε ένα απλό παράδειγμα της χρήσης της βιβλιοθήκης της **οθόνης LCD**, ώστε να γίνει αντιληπτός ο τρόπος με τον οποίο δηλώνουμε της θύρες του Arduino και της οθόνης αλλά και οι συναρτήσεις της βιβλιοθήκης. Το ποτενσιόμετρο χρησιμοποιείται για την ρύθμιση της αντίθεσης των χαρακτήρων.



Εικόνα: Συνδεσμολογία της οθόνης LCD,



Εικόνα: Σχηματικό διάγραμμα της οθόνης LCD

```
#include <LiquidCrystal.h>      //  δήλωση της βιβλιοθήκης
LiquidCrystal lcd(12,    11, 5, 4, 3, 2); // δηλώνουμε τα PIN της οθόνης
                 //(rs,enable,d4,d5,d6, d7)
 void setup(){                {
  lcd.begin(16, 2); // ορίζουμε τις στήλες και τις γραμμές της οθόνης
  lcd.print("hello,world!");} // εκτυπώνουμε το μήνυμα «hello,world»
             }

void loop(){
  lcd.setCursor(0, 1);} //θέτουμε τον κέρσορα στην επιθυμητή θέση (1η στήλη, 2η γραμμή
  lcd.print(millis()/1000); // εκτυπώνουμε τα δευτερόλεπτα από τηντελευταία επαναφορά
           }
```

Μηχατρονικά συστήματα ΙΙ

Ο μικροελεγκτής «Arduino Uno» και τα εξωτερικά «Interrupt»

Η υπολογιστική πλατφόρμα Arduino Uno χρησιμοποιεί μέχρι και δύο εξωτερικά interrupt, τα **INT0** και **INT1**, τα οποία εκ κατασκευής είναι ορισμένα στις **θύρες** (PIN) **2 και 3** αντίστοιχα. Το εξωτερικό interrupt είναι ένα σήμα που δέχεται ο μικροελεγκτής οποιαδήποτε χρονικά στιγμή, για ένα γεγονός το οποίο χρειάζεται άμεση προσοχή και όταν ένα interrupt ενεργοποιηθεί ο μικροελεγκτής σταματά οποιαδήποτε εργασία πράττει και εκτελεί την ρουτίνα του interrupt (interrupt service routine). Μετά την εκτέλεση της ρουτίνας του interrupt η εκτέλεση του προγράμματος συνεχίζει από την εντολή στην οποία προέκυψε η διακοπή. Τα interrupt είναι πολύ χρήσιμα καθώς μας επιτρέπουν να εκτελούμε μια συνάρτηση άμεσα και ταχύτατα. Για να χρησιμοποιήσουμε ένα εξωτερικό interrupt αρκεί να χρησιμοποιήσουμε την συνάρτηση π.χ. attachInterrupt στην οποία πρέπει να δηλώσουμε τον αριθμό του interrupt ,την συνάρτηση του interrupt και τον τρόπο λειτουργίας του interrupt, δηλαδή **attachInterrupt (interrupt, function_name, mode).** Προφανώς στη θέση του interrupt επιλέγουμε ανάμεσα από το **INT0** και **INT1**, η συνάρτηση του interrupt (interrupt service routine) καλείται κάθε φορά που ενεργοποιείται το interrupt, δεν χρειάζεται κάποια παράμετρο και δεν επιστρέφει τίποτα. Στη θέση του **function_name** γράφουμε το όνομα της συνάρτησης του interrupt, ενώ όσον αφορά τον τρόπο λειτουργίας (**mode**) του interrupt υπάρχουν τέσσερις επιλογές οι οποίες είναι:

- **LOW** : το interrupt ενεργοποιείται όταν στη θύρα (PIN) που είναι συνδεδεμένο το interrupt υπάρξει σήμα LOW.
- **CHANGE** : το interrupt ενεργοποιείται όταν στη θύρα που είναι συνδεδεμένο το interrupt υπάρξει αλλαγή στο σήμα οποιαδήποτε και αν είναι αυτή.
- **RISING** : το interrupt ενεργοποιείται όταν έχουμε ανερχόμενο μέτωπο δηλαδή, όταν στη θύρα που είναι συνδεδεμένο το interrupt υπάρξει σήμα που μετατρέπεται από LOW σε HIGH.
- **FALLING**: το interrupt ενεργοποιείται όταν έχουμε κατερχόμενο μέτωπο δηλαδή, όταν στη θύρα που είναι συνδεδεμένο το interrupt υπάρξει σήμα που μετατρέπεται από HIGH σε LOW.

Τέλος, μέσα στη συνάρτηση του interrupt η συνάρτηση **delay()** δεν λειτουργεί, οι τιμές της εντολής **millis()** δεν αυξάνονται. Οι μεταβλητές που χρησιμοποιούνται μέσα και έξω από τη συνάρτηση του interrupt πρέπει να είναι μορφής **volatile**.

Στον παρακάτω πίνακα παρουσιάζονται διάφορα μοντέλα μικροελεγκτών με τις αντίστοιχες ακίδες των interrupts.

Board	Digital Pins Usable For Interrupts
Uno, Nano, Mini, other 328-based	2, 3
Mega, Mega2560, MegaADK	2, 3, 18, 19, 20, 21
Micro, Leonardo, other 32u4-based	0, 1, 2, 3, 7
Zero	all digital pins, except 4
MKR1000 Rev.1	0, 1, 4, 5, 6, 7, 8, 9, A1, A2
Due	all digital pins

Παράδειγμα χρήσης των εξωτερικών Interrupt («Interrupt Service Routine»)

Η χρήση των εξωτερικών Interrupt («Interrupt Service Routine») γίνεται αντιληπτή με το παρακάτω παράδειγμα, με το οποίο το LED 13, που βρίσκεται πάνω στην **υπολογιστική πλατφόρμα του Arduino**, εναλλάσσεται η κατάσταση του («HIGH-LOW») κάθε φορά που ενεργοποιείται το Interrupt 0 (INT0) .

```
// Ακολουθούν  η συνάρτηση αρχικών ρυθμίσεων του μικροελεγκτή
int pin = 13; //δήλωση του LED
volatile int state = LOW;//δήλωση της μεταβλητής state


// Ακολουθεί η συνάρτηση αρχικών ρυθμίσεων του μικροελεγκτή
void setup(){
       pinMode(pin, OUTPUT); //δήλωση του LED ως έξοδο
       attachInterrupt(0, blink, CHANGE); //δήλωση της συνάρτησης του
            }                                  //interrupt (INT0) στο Pin2

// Ακολουθεί η κύρια ρουτίνα του προγράμματος που εκτελείται συνεχώς
void loop(){
       digitalWrite(pin, state); //εγγραφή της κατάστασης του LED
           }
// Ακολουθεί η Ρουτίνα εξυπηρέτησης της διακοπής
void blink() {
          state = !state; //αναστροφή της κατάστασης του LED
             }
```

Παράδειγμα 3: Άναμμα και σβήσιμο Led ελεγχόμενο από εξωτερικό button με την διαδικασία Interrupt (Η διαφορά με την προηγούμενη περίπτωση είναι ότι πρίν η ενεργοποίηση γίνεται μετά το loop ενώ τώρα άμεσα με την ενεργοποίηση της διακοπής)

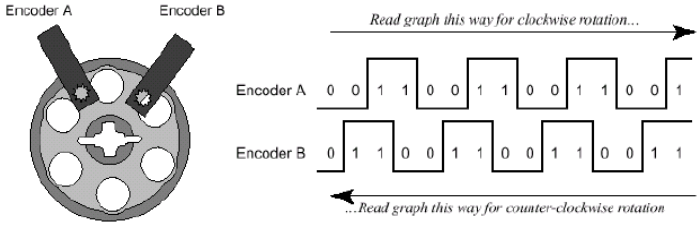
```
const int LED = 13; // Ορίζουμε μία σταθερά τύπου Integer (Ακεραίου) με όνομα
                    //LED και της δίνουμε την τιμή 13 με σκοπό να την
                    //χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε που
                    //θα συνδέσουμε το led.
 const int BUTTON = 9; // Ορίζουμε μία σταθερά τύπου Integer (Ακεραίου) με όνομα
                    // BUTTON και της δίνουμε την τιμή 9 με σκοπό να την
                    // χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε
                    // που θα συνδέσουμε το button.
// Ακολουθεί η συνάρτηση αρχικών ρυθμίσεων του μικροελεγκτή
void setup () {
        pinMode (LED, OUTPUT); // Ορίζουμε χρησιμοποιώντας την μεταβλητή LED,
                               // ότι το pin 13 θα λειτουργεί σαν ψηφιακή
                               // έξοδος 0-5V

         pinMode (BUTTON, INPUT);
     digitalWrite (BUTTON, HIGH); //ενεργοποιώ την internal pull-up resistor
                                  // Ορίζουμε χρησιμοποιώντας την μεταβλητή
                                  // BUTTON, ότι το pin 9
                                  // θα λειτουργεί σαν ψηφιακή είσοδος 0-5V
   attachInterrupt (0, ISRpinChange, CHANGE);// Ενεργοποίηση της ρουτίνας
                                             // που θα εξυπηρετεί την
                                             // διακοπή. Η πρώτη παράμετρος
                                             // με τιμή 0 (0 => INT0=> pin 2)μας
                                             // ορίζει την πηγή της εξωτερικής
                                             //διακοπής (για τιμή 1=> INT1=> pin 3).
                                             //Η δεύτερη παράμετρος με
                                             // τιμή ISRpinChange μας ορίζει το
                                             // όνομα της ρουτίνα εξυπηρέτησης
                                             // της διακοπής ISR που δημιουργήσαμε
                                             // παραπάνω. Η τρίτη παράμετρος με
                                  // τιμές LOW, RISING, FALLING, CHANGE μας ορίζει το
                                  // είδος της αλλαγής λογικής στάθμης που ενεργοποιεί
                                  // την διακοπή.
                }
// Ακολουθείται η κύρια ρουτίνα του προγράμματος που εκτελείται συνεχώς
void loop () {
                // Δεν εκτελείται καμία εντολή αρά δεν γίνεται τίποτα και
                // περιμένουμε να συμβεί μία  διακοπή
               }

// Ακολουθεί η Ρουτίνα εξυπηρέτησης της διακοπής
// Interrupt Service Routine (ISR)
// Ακολουθεί η δημιουργία ρουτίνας εξυπηρέτησης διακοπής με όνομα pinChange (μπορεί να
δηλωθεί μετά τις μεταβλητές και πριν την ρουτίνα αρχικών ρυθμίσεων του μικροελεγκτή ή και
μετά την συνάρτηση void loop() όπως κάνουμε εδώ)

void ISRpinChange(){
            noInterrupts (); // Απενεργοποίηση των υπόλοιπων διακοπών με σκοπό
                             // να μην συμβεί κάποια άλλη διακοπή για όσο χρόνο
                             // εξυπηρετείται η παρούσα διακοπή. Αν έχουμε μόνο
                             // μία ISR τότε δεν απαιτείται να γράψουμε αυτήν την
                             // εντολή. Επίσης μέσα στις ρουτίνες ISR δέν
                             //επιτρέπεται η χρήση της συνάρτησης delay().
          if (digitalRead (BUTTON) == HIGH) // Διαβάζουμε την ψηφιακή τιμή
                                            // που έχει το Pin 9 (εκεί που
                                            // συνδέεται το button) και
                                            // ελέγχουμε αν το button είναι
                                            // πατημένο, αν ναι τότε
              digitalWrite (LED, HIGH); // Βγάλε λογικό 1 (5V) στο pin 13,
                                        // δηλαδή άναψε το led
           else digitalWrite (LED, LOW);//Αλλιώς: Βγάλε λογικό 0
```

```
                                                //(0V) στο pin 13, δηλαδή σβήσε το led
            interrupts (); // Ενεργοποίηση των διακοπών
                }
```

Στην συνέχεια θα παρουσιάσουμε τον κινητήρα που χρησιμοποιείται στο εμπορικά διαθέσιμο πακέτο της LEGO και θα χρησιμοποιήσουμε τον οπτικό κωδικοποιητή που διαθέτει ο κινητήρας προκειμένου με την βοήθεια των interrupts να υπολογίζουμε την γωνία της περιστροφικής άρθρωσης. Ας ξεκινήσουμε όμως βλέποντας από τι αποτελείται ο κινητήρας.

Περιγραφή του κινητήρα «NXT» της Lego

Ο κινητήρας **NXT** της Lego έχει ενσωματωμένο περιστροφικό κωδικοποιητή («rotary encoder») δύο καναλιών που αποτελείται από ένα στρεφόμενο δίσκο που είναι διαιρεμένος σε τομείς οι οποίοι είναι εναλλάξ διαφανείς και μη-διαφανείς. Στη μια πλευρά του δίσκου υπάρχει μια φωτεινή πηγή και στην άλλη ένα φωτοκύτταρο. Όταν ο δίσκος περιστρέφεται, κάθε μεταβολή του φωτός που προσπίπτει στο φωτοκύτταρο παράγει ένα παλμό εξόδου. Το πλήθος των παλμών αυτών ανά μονάδα χρόνου είναι ανάλογο προς τη γωνιακή ταχύτητα του άξονα και το ολικό πλήθος παλμών σε κάθε χρονική στιγμή είναι ανάλογο προς την ολική γωνιακή μετατόπιση του μετρούμενου άξονα. Ο κωδικοποιητής έχει δύο φωτοκύτταρα ανίχνευσης στον ίδιο δίσκο, τα οποία είναι τοποθετημένα κατά τέτοιο τρόπο, ώστε οι έξοδοι να έχουν διαφορά φάσης 90° η μία, σε σχέση με την άλλη. Η φορά περιστροφής προσδιορίζεται με τη βοήθεια κατάλληλου λογικού κυκλώματος, το οποίο δέχεται σαν εισόδους τις δυο ακολουθίες παλμών, όπως φαίνεται και στο παρακάτω σχήμα (Εικ.3.4.1).

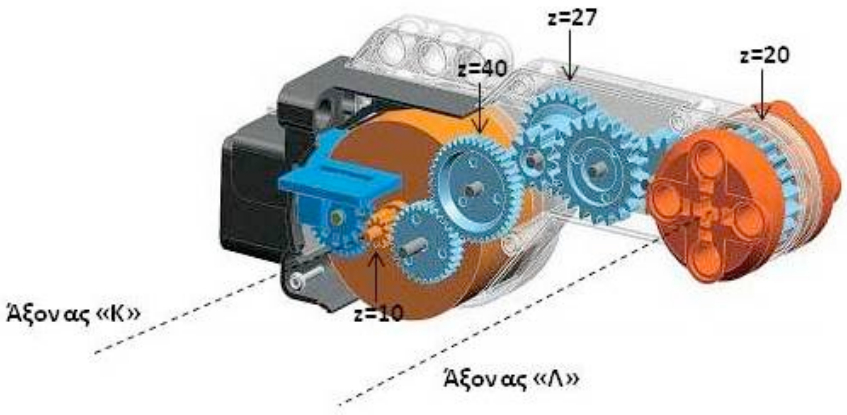


Εικόνα 3.4.1 Απεικόνιση των καναλιών (A, B) του περιστροφικού κωδικοποιητή του κινητήρα NXT

Από την έξοδο του κινητήρα η **σχέση μετάδοσης**, δηλαδή ο λόγος των διαμέτρων συνεργαζόμενων γραναζιών εσωτερικά του κινητήρα μπορεί να βρεθεί με βάση την εικόνα (Εικ.3.3.2.3) και τον παρακάτω πίνακα ο οποίος περιγράφει την σχέση των γραναζιών.

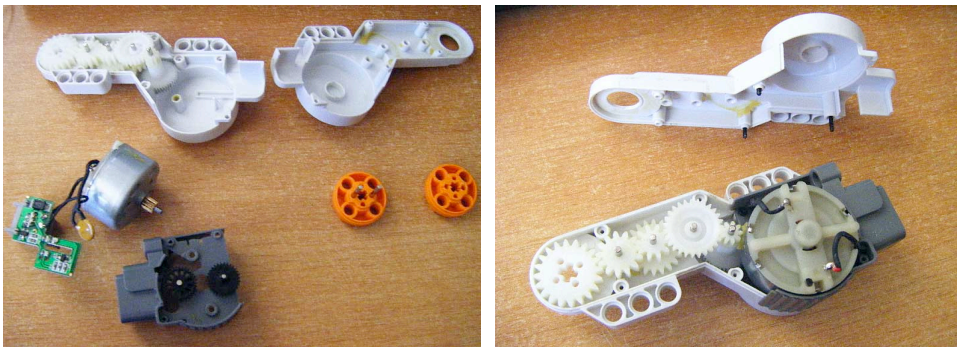
10:30:40	= 1:4
9:27	= 1:3
10:20	= 1:2
10:13:20	= 1:2
Overall	1:48

Πίνακας 3.4.2



Εικόνα 3.4.3 Εσωτερική όψη του κινητήρα "NXT"

Προκύπτει από τον παραπάνω πίνακα (Πίν.3.4.2) ότι η **σχέση μετάδοσης** είναι 1:48 δηλαδή για μια περιστροφή του άξονα «Λ» (hub) ο άξονας «Κ» πρέπει να κάνει 48 περιστροφές. Η **σχέση μετάδοσης** μεταξύ του γραναζιού στο οποίο βρίσκεται ο κωδικοποιητής (z=32) και του γραναζιού(z=10) που βρίσκεται πάνω στον άξονα «Κ» είναι 10:32, άρα για μία περιστροφή του άξονα «Λ»(hub) ο κωδικοποιητής κάνει 48*10:32=15 περιστροφές.

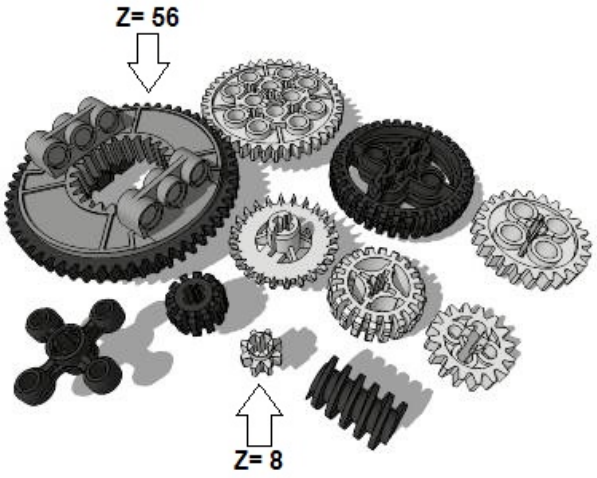


Εικόνα 3.4.4 Αποσυναρμολογημένη όψη του κινητήρα NXT

Ο κωδικοποιητής του **NXT** έχει 12 σχισμές, οπότε οι οπτικοί ανιχνευτές «βλέπουν» 12*15=180 παλμούς, χρησιμοποιώντας και τα δύο κανάλια του κωδικοποιητή έχουμε 360 παλμούς ανά περιστροφή και καθώς **ο κωδικοποιητής του NXT είναι τετραπεριοδικής διεύθυνσης («quadrature encoder») το μέγιστο πλήθος παλμών που μπορεί να παράγει ο αποκωδικοποιητής είναι 720 παλμοί ανά περιστροφή**. Χρησιμοποιώντας το γρανάζι των 8 δοντιών στην έξοδο του κινητήρα (άξονας «Λ»(hub)) του εκπαιδευτικού πακέτου Lego, και στον βραχίονα αυτό των 56 δοντιών, η μεταξύ τους σχέση μετάδοσης είναι η εξής:

$$i = \frac{z_1}{z_2} = \frac{8}{56} = \frac{1}{7} = 0.14$$

*όπου **i** η σχέση μετάδοσης των γραναζιών, **z1** ο αριθμός δοντιών του κινητήριου τροχού και **z2** ο αριθμός δοντιών του κινούμενου τροχού*[14].

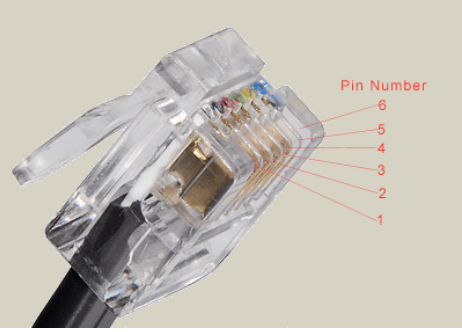


Εικόνα *3*.4.5 Τα γρανάζια του εκπαιδευτικού πακέτου Lego Mindstorms *NXT*

Όπως και όλα τα υπόλοιπα μέρη του πακέτου **Lego Mindstorms NXT**, ο κινητήρας χρησιμοποιεί για την σύνδεση του το καλώδιο «RJ12», το οποίο είναι ένα διεθνές πρότυπο σύνδεσης σταθερών τηλεφώνων και άλλων παρόμοιων εφαρμογών. Έτσι για την σύνδεση του με την **υπολογιστική πλατφόρμα Arduino Uno** χρειάστηκε να αφαιρέσουμε την μια άκρη του καλωδίου ώστε να συνδέσουμε το κάθε καλώδιο στις θύρες (pin) του Arduino. Την ίδια διαδικασία ακολουθήσαμε σε όλα τα μέρη του πακέτου της Lego που χρησιμοποιούν το συγκεκριμένο καλώδιο(«RJ12»). Παραθέτουμε παρακάτω τον πίνακα με τις πληροφορίες για το καλώδιο «RJ12».

Επεξήγηση των ακροδεκτών του καλωδίου «RJ12»

Θύρες/Pin	Ονομασία	Χαρακτηριστικά καλωδίου	Χρώμα καλωδίου	Αρίθμηση Θυρών/Pin
1	ANALOG	+9V Τροφοδοσία	Άσπρο	
2	GND	Γείωση	Μαύρο	
3	GND	Γείωση	Κόκκινο	
4	IPOWERA	+5V Τροφοδοσία	Πράσινο	
5	DIGIAI0	Κανάλι Α οπτικού κωδικοποιητή	Κίτρινο	
6	DIGIAI1	Κανάλι Β οπτικού κωδικοποιητή	Μπλε	



Εικόνα 3.4.6 Καλώδιο «RJ12»

Όπως φαίνεται στη παραπάνω πίνακα η συνδεσμολογία για το καλώδιο «RJ12» είναι η εξής:

- Το **άσπρο** καλώδιο(Pin 1) είναι η τροφοδοσία (9V)
- Το **μαύρο** καλώδιο(Pin 2) η γείωση
- Το **κόκκινο** καλώδιο(Pin 3) η γείωση του αισθητηρίου(εάν υπάρχει)
- Το **πράσινο** καλώδιο(Pin 4) η τροφοδοσία (4.3V) του αισθητηρίου(εάν υπάρχει)
- Το **κίτρινο** καλώδιο(Pin 5) το κανάλι (B)
- Το **μπλε** καλώδιο(Pin 6) το κανάλι (A)

Με το παρακάτω πρόγραμμα μπορούμε να διαβάσουμε τον encoder του κινητήρα και να τυπώσουμε στην σειριακή θύρα του υπολογιστή την γωνία περιστροφής του κινητήρα αξιοποιώντας τα INT0 και INT1 του μικροελεγκτή Arduino UNO. Για περισσότερες πληροφορίες μπορείτε να ανατρέξετε στην διεύθυνση:

http://playground.arduino.cc/Main/RotaryEncoders#Example3

```
          #define encoder0PinA 2
          #define encoder0PinB 3
 volatile unsigned int encoder0Pos = 32768;
          unsigned int tmp = 0; //temporary position
          unsigned int Aold = 0;
          unsigned int Bnew = 0;
volatile double angle = 0; // motor's angle
void setup() {
        pinMode(encoder0PinA, INPUT);
        pinMode(encoder0PinB, INPUT);
attachInterrupt(0, doEncoderA, CHANGE); // encoder pin on interrupt 0 (pin 2)
attachInterrupt(1, doEncoderB, CHANGE); // encoder pin on interrupt 1 (pin 3)
  Serial.begin (115200);}

void loop(){//Check each changes in position
            if (encoder0Pos != tmp) {
                angle = (360.0/720.0)*(encoder0Pos-32768); Serial.println(angle);
               Serial.print("\t");
             Serial.println(encoder0Pos, DEC);
                  tmp = encoder0Pos;}
delay(10);}

void doEncoderB(){// Interrupt on B changing state
  Bnew=digitalRead(encoder0PinB);
  Bnew^Aold ? encoder0Pos++:encoder0Pos--;}

void doEncoderA(){ // Interrupt on A changing state
  Bnew^Aold ? encoder0Pos++ : encoder0Pos--; //Ternary operator (condition ? Evaluated_when_true : evaluated_when_false)
  Aold=digitalRead(encoder0PinA);}
```

για την καλύτερη κατανόηση του παραπάνω προγράμματος παραθέτουμε το παρακάτω σχήμα το οποίο αναπαριστά την κεντρική ιδέα του αλγορίθμου των interrupts.